Benchmark

	Expansions			Goal tests			New nodes			Plan length			Elapsed time (seconds)		
	P1	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**
breadth_first_search	43	3343	14663	56	4609	18098	180	30509	129631	6	9	12	0.05	21.6	154.57
breadth_first_tree_search	1458			1459			5960			6			1.5		
depth_first_graph_search	21	624	408	22	625	409	84	5602	3364	20	619	392	0.02	4.77	3.56
depth_limited_search	101	222719		271	2053741		414	2054119		50	50		0.14	1018.3	
uniform_cost_search	55	4852	18234	57	4854	18236	224	44030	159707	6	9	12	0.07	66.46	352
recursive_best_first_search	4229			4230			17023			6			5.1		
greedy_best_first_graph_search	7	990	5605	9	992	5607	28	8910	49360	6	17	22	0.009	11	170.72
astar_search_h_1	55	4852	18234	57	4854	18236	224	44030	159707	6	9	12	0.07	61.2	355.12
astar_search_h_ignore_preconditions	41	1506	5118	43	1508	5120	170	13820	45650	6	9	12	0.06	16.07	124.4
astar_search_h_pg_levelsum	11	86	408	13	88	410	50	841	3758	6	9	12	3.12	460.9	4458

Optimal Plan

Problem 1	Problem 2	Problem 3
Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO)	Load(C1, P1, SFO) Load(C2, P2, JFK) Load(C3, P3, ATL) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Fly(P3, ATL, SFO) Unload(C3, P3, SFO)	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Unload(C3, P1, JFK) Fly(P2, ORD, SFO) Unload(C2, P2, SFO) Unload(C4, P2, SFO)

Analysis

BFS found optimal solution for all 3 problems. This is understandable since the path cost for this problem is non decreasing, in other words there is no node with negative cost. BFS, however, also tend to use up a lot more memory and computational power due to its nature of holding all frontier nodes in memory to examine before returning the solution.

Uniform-cost search is comparable to BFS performance-wise and could also find optimal solutions for all 3 problems, since it's similar to BFS but instead of using a stack to store the frontiers, it used a priority queue, which makes it inherit the same heavy memory usage weakness from BFS, but also ensures that it can find the optimal solution.

DFS always finish but never give optimal solution. The DFS solutions have redundant actions such Load followed by Unload which makes it inferior to other solutions. DFS, however, is more efficient in terms of memory usage - less nodes expanded in general since it goes straight deep down the search tree to return the first goal found - which makes it more likely to return suboptimal goals.

Among A* heuristics, the levelsum heuristic is always more efficient in terms of nodes expanded, however it also finish lasts, due to the heavy task of recreating the planning graph every time. Also, the A* algorithms only show some advantages from P3, since for smaller problems like P1 and P2, the cost of estimating the heuristics simply outweighs its benefits.

Given a constant heuristic, A* performs just like uniform cost search. This is understandable since the only difference between them is the heuristic, without that every nodes' potential cost are treated equally and path cost is the only thing that affects the decision.

Best heuristic:

When the goal is just to find a path with optional optimality, DFS is the best candidate since it's both complete and use less resources.

When optimality is a must, we can consider A* with either ignore_preconditions or level_sum heuristics, since they both directs the solver towards to optimal path. If we only care about the expanded nodes, level_sum is the best candidate. But in practice, ignore_preconditions have shown superiority thanks to the lack of computational overhead which makes level_sum yield a much worse runtime.